HOSTED BY

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/jides

Boundary and holes recognition in wireless sensor networks

*Rachid Beghdad**, *Amar Lamraoui*

Faculty of Sciences, Abderrahmane Mira University, Béjaïa 06000, Algeria

HIGHLIGHTS

- A distributed solution for detecting boundaries and holes in the Wireless Sensor Network (WSN) is proposed.
- At first, each node collects connectivity information of its one-hop neighbors and constructs its one-hop neighbors' graph.
- Secondly, independent sets are constructed.
- Finally, the independent sets are connected in order to find the closed path.
- Our algorithm performs better than some former works.

ARTICLE INFO

Article history:
Published online 3 May 2016

Keywords:
Boundary recognition
Holes detection
Wireless sensor networks
Independent set
Graph theory

ABSTRACT

In this paper, a distributed solution is proposed for detecting boundaries and holes in the WSN using only the nodes connectivity information. The run of our protocol is divided imto three main steps. In the first step, each node collects connectivity information of its one-hop neighbors and constructs its one-hop neighbors' graph. In the second step, independent sets are constructed. In the last step, the independent sets are connected in order to find the closed path. Therefore, the node can make its own decision to be an internal or a boundary node. Simulation results show that our algorithm can detect fine-grained boundaries with high accuracy, low energy consumption and less communication overhead compared to some former works. In addition, this algorithm performs better than some exiting approaches (BCP, THD, and SDBR).

1. Introduction

Due to the random deployment of sensor nodes, nodes' failure, or an environmental obstacle (building, lake . . .) holes can be formed in the network, creating sets of isolated nodes and leaving uncovered areas. Moreover, they can also cause the failure of routing algorithms. Once detecting either the nodes on the boundaries of holes or on the network's boundary; uncovered areas will be detected and could be repaired by an incremental addition of new sensors, the aforementioned detection also allows the routing protocols to identify and pass these holes [1].

The existing boundary detection algorithms can be classified into three main categories according to their used

Peer review under responsibility of Qassim University.

* *Corresponding author.*
E-mail addresses: rachid.beghdad@gmail.com (R. Beghdad), amar.lamraoui@gmail.com (A. Lamraoui).

http://dx.doi.org/10.1016/j.jides.2016.04.001

techniques: geometrical methods, statistical methods and topological methods. Each category has its advantage and its weaknesses.

This paper aims to provide a simple and efficient distributed approach to discover the boundary nodes in the network. We distinguish two types of boundary nodes: the internal boundary nodes that surround the holes and the outer boundary nodes, which lie on the external boundary of the sensing field [2]. Regarding their small size, low cost and limited energy, most of the sensors are not equipped with a positioning device. Therefore, we rely solely on the topology extracted from the available connectivity information.

The run of our Boundary Detection protocol based on Connected Independent Sets (BDCIS) is divided in three main steps. In the first step, each node collects connectivity information of its one-hop neighbors and constructs its one-hop neighbors' graph. In the second step, independent sets of cardinality α are established. In the last step, the independent sets are connected in order to search for the closed path. Therefore, each node can make its own decision to be internal or boundary node.

The rest of this paper is organized as follows:
Section 2 gives an overview of some existing boundary detection algorithms. Section 3 presents our alternative solution for the stated problem. Simulations and performances' analysis are given in Section 4. Finally, Section 5 concludes the paper and brings some future perspectives.

2. Related work

The existing boundary detection algorithms can be classified into three main categories according to their techniques: geometrical methods, statistical methods and topological methods [3].

Geometrical methods assume that nodes are aware of their geographical positions by using GPS or other positioning device. These methods can find boundaries with high accuracy and less control messages.

Fang et al. [4] studied a fundamental problem behind the "local minimum phenomenon" in geographic forwarding. They defined the stuck nodes where packets can possibly get stuck in greedy multi-hop forwarding, and developed a local rule, the TENT rule, for each node in the network to test if it is a stuck node. To help packets get out of the stuck nodes, they developed a distributed algorithm, BOUNDHOLE, to find the so-called holes, the regions of the network with boundaries consisting of all the stuck nodes. Holes are usually associated with regions where nodes are depleted or regions that do not have enough nodes due to irregular terrain. Holes have sometimes been referred to as "communication voids" as well. Both their analysis and simulations show that the holes identified using this method indeed capture the underlying structure of the network and correctly identify regions with communication voids. Another centralized algorithm proposed in [4], which is based on Restricted Delaunay Graph (RDG). Hole is defined to be a face in the RDG with at least four vertices.

Sahoo et al. [5] proposed sequential and distributed boundary nodes selection SBNS and DBNS algorithms. The SBNS algorithm assumes the sink to be a boundary node then uses the right hand rule to select boundary nodes in a sequential manner. The process is lunched by the sink and is repeated until the starting node (sink) is revisited. The DBNS algorithm defines extreme nodes as boundary nodes then connecting them to form cycles enclosing boundaries. An extreme node is defined as a node that has either maximum or minimum value in its coordinates compared to of its one-hop neighbors. The main drawback of these methods is the need of an accurate coordinates of sensor nodes. Each node must be equipped with a positioning device such as GPS to obtain its geographical location, which is not suitable for small sensors with low energy consumption.

Fekete et al. [6] described a new approach for dealing with the central problem in the self-organization of a geometric sensor network: given a polygonal region R, and a large, dense set of sensor nodes that are scattered uniformly at random in R. There is no central control unit, and nodes can only communicate locally by wireless radio to all other nodes that are within communication radius r, without knowing their coordinates or distances to other nodes. The objective is to develop a simple distributed protocol that allows nodes to identify themselves as being located near the boundary of R and form connected pieces of the boundary. They gived a comparison of several centrality measures commonly used in the analysis of social networks and show that restricted stress centrality is particularly suited for geometric networks; they provided mathematical as well as experimental evidence for the quality of this measure. Fekete et al. [7] also proposed another boundary detection algorithm called Connectivity-based Distributed Coverage Hole Detection CDCHD. The basic idea is that nodes on the boundaries have relatively smaller average degrees than nodes inside the network. A statistical threshold is used to distinguish between boundary nodes and internal nodes.

Statistical methods assume that the distribution of nodes within the network follows some statistical properties.

Destino [8] suggested a centralized algorithm Boundary Recognition via Graph-Theory (BRGT) based on a graph clustering technique, which allows the division of the network into small cells (clusters) that circumvent connection holes. Then, the boundary nodes of each cluster are identified using centrality scores. The detection of boundary nodes is performed by the fusion of adjacent clusters. Only nodes that are on the border of a single cluster and are not connected to two or more clusters are selected as boundary nodes. The algorithm is centralized and the entire network topology must be collected by the base station to begin the discovery process, which is not suitable for large sensor networks.

A Topological Hole Detection (THD) algorithm based on the idea of using the iso-contours was introduced by Funke [9]. After choosing four beacons, the nodes having the same hop count from a beacon should belong to the same contour. These contours are broken into connected components due to the presence of holes or when they meet the outer boundary. For each connected component, a local beacon is chosen and the computing of shortest distance from this beacon is performed. The nodes with highest distance values must lie on both 'ends' of each connected component. These nodes are marked as boundary nodes. This

algorithm is implemented in a distributed manner in [10] by selecting a maximal independent set of nodes in the network (seeds). Then iso-contours are constructed for (6) six-hop neighborhood around each seed node.

Topological methods use only the available connectivity information to detect boundaries in the network.

In his distributed protocol, Wang et al. [3] built a shortest path tree through flooding the whole network starting from a seed node. Then the algorithm searches for cut nodes that are defined as nodes which have their Least Common Ancestor (LCA) relatively far away and their paths to the LCA well separated. By using the cut nodes, the algorithm artificially merges holes into a single hole to construct one composite cycle R, then, it synchronizes the nodes of R and floods the whole network to find the outer boundary. This algorithm has been criticized for the flooding of the whole network and the synchronization of the nodes of the cycle R, which requires more execution time and communication overhead.

In [2], Hseih et al. described the Distributed Boundary Recognition Algorithm (DBRA) which is based on four phases. The first phase is to select Closure Nodes (CNs) which roughly enclose the holes and frontier of the sensing field. It is to be noted that, for simplification, the authors used the term obstacles instead of the holes and frontier of sensing field. In the second phase, those closure nodes are connected with each other to form Coarse Boundary Cycles (CBCs) for identifying each obstacle. The third phase is proposed to discover the exact Boundary Nodes (BNs) and connect them to refine the CBCs to be final boundaries. The last phase is proposed to maintain the integrity of BNs while the boundary is broken due to nodes failure.

The drawback of this approach is the large amount of exchanged messages, energy consumption and selection time.

Khan et al. [11] introduced a Self-Detection algorithm for Boundary Recognition (SDBR). Every node can decide to be a boundary or internal one by the construction of its 2-hop neighbors' graph. Then the node will checks if this graph forms a closed cycle or broken path. The broken path indicates the node is residing on the boundary, while in case of the closed cycle the node is marked as internal node. The major lack of this algorithm is its selection of coarse boundaries in which several internal nodes are wrongly recognized as boundary nodes.

Dabba [12] proposed a Border Coverage Protocol (BCP) based on three phases: the first one consists to detect the nodes close to the border and that will be considered as boundaries nodes. The second phase is dedicated to find the transfer and internal nodes. The last phase focused on replacing border nodes by the transfer nodes, in case of any failure. In this algorithm, boundaries nodes are detected using Localized Voronoï Polygons (LVP). This algorithm assumes that each sensor node has four sides, if the node receives at least one message from each side among the four, its LVP is finite, thus, the node will decide to be internal node. In the other case, the LVP is infinite and the node sets its self to be boundary node. The main inconvenience of this algorithm is the awareness of the four sides of each node.

Table 1 summarizes the above cited approaches.

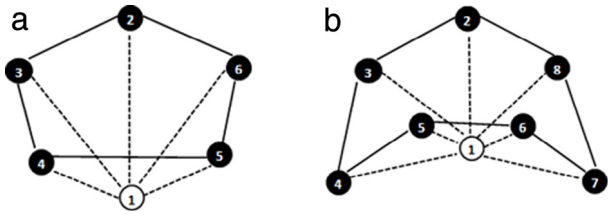


Fig. 1 – Examples of configurations causing false detections by Hamiltonian cycles.

2.1. *Discussion*

RDG [4] is a centralized approach, thus, it does not scale with large wireless networks, in terms of running time, and communication overhead. Moreover, it requires the exact coordinates of sensors, which is not available in all WSNs. Fekete et al. [7] approach can detect only BNs that enclose holes and cannot detect the outer boundary. The rate of false detection increases with the density of the network, and the rate of messages overhead increases proportionally with the number of holes.

The main disadvantage of geometrical algorithms [4–7] is the need of uniform distribution and a high average node degree.

The main drawback of statistical algorithms [8–10] is the recurring flooding of the whole network to select the global beacons, in addition to the process of selecting local beacons, which incurs considerable cost in terms of time consumption and communication overhead.

The main drawback topological algorithms [2,11,12] are respectively: the large amount of exchanged messages, energy consumption and selection time, selection of coarse boundaries in which several internal nodes are wrongly recognized as boundary nodes, and is the awareness of the four sides of each node.

This is the reason why, the aim of our paper is to propose a new algorithm for boundary detection in WNSs that can avoid the above cited drawbacks.

3. The proposed boundary detection algorithm

We propose an alternative solution to detect holes as well as the outer boundary of the wireless sensor network using only connectivity information.

An internal node is covered by its neighbors that surround it by all its sides, and can form a closed cycle around it. This cycle can be broken for a boundary node, if it has at least an uncovered portion. Due to the random distribution and network density, finding this cycle is highly difficult even Hamiltonian cycles are used in [13]. Fig. 1 depicts an example of configurations causing false detections in the one hop neighbors' graph. Despite the node "1" is a boundary node; we can find a Hamiltonian cycle in its one hop neighbors' graph.

The problem is how to select a sub set of nodes among all the neighbors, then to examine the presence of a cycle to determine if the current node is an internal or a boundary node.

Table 1 – A set of existing approaches for detecting boundary nodes.

Protocol	Location awarness	Additional information	Protocol type	Network flooding	Boundary detection
RDG [4]	Yes	Coordinates	Centralized	The whole network	Holes
DBNS [5]	Yes	Coordinates	Distributed	Boundary nodes	Network + Holes
SBNS [5]	Yes	Coordinates	Distributed	Boundary nodes	Network
CDCHD [7]	No	Distances	Distributed	2-hop neighbors	Holes
BRGT [8]	No	Angles	Centralized	The whole network	Network + Holes
THD [9]	No	Distances	Centralized	The whole network	Network + Holes
LCA [3]	No	Distances	Distributed	The whole network	Network + Holes
CBC [2]	No	Distances	Distributed	The whole network	Network + Holes
SDBR [11]	No	Distances	Distributed	3 hop neighbors	Network + Hole
BCP [12]	No	Angles	Distributed	1 hop neighbor	Network + Hole

Our main contribution is the use of the independent sets' concept from graph theory [14] in which every node will select a subset among its neighbors, connecting them taking into consideration some rules to construct a closed path. To select a suitable set of neighboring nodes to be examined, these nodes must be quite far away from each other, where these nodes are distributed in all sides of the node. The best way to select this set is the use of independent sets. The connection of these chosen nodes provides a ring that encircles the internal nodes, while this ring is broken for the boundary nodes.

3.1. *Network model and assumptions*

The network is modeled by a communication graph denoted $G = (V, E)$, where V (vertices) represents the set of sensor nodes and E (edges) denotes the communication links between them. A link is established if two nodes can communicate with each other.

- Each node in the network has a unique identifier (Id_i).
- The network is assumed to be connected with symmetrical links, sensors are static and there are no transmission errors.
- All nodes within the network have the same communication range and the communication graph follows the unit disc graph model (*UDG*). Thus, a node n_i can communicate with another node n_j if the distance between them is less than the communication range C_R.
- The set of neighbors $N(u)$ of a node u is defined by:

$$N(u) = \{v \in V : v \neq u \Lambda (u, v) \in E\} \text{ [11]}. \quad (1)$$

- $N_k(u)$ is the *k-hop* neighbors' set of the node u. It consists of all nodes which the less number of hops to u are equal to k.
- The degree of a node x is equal to the number of nodes within its communication range C_R [11].

3.2. *BDCIS description*

The run of our Boundary detection algorithm based on connected independent sets (BDCIS) is divided into three main steps. In the first step, each node collects connectivity information of its one-hop neighbors and constructs its one-hop neighbors' graph. In the second step, independent sets of cardinality α are established. In the last step, connecting sets and searching for closed paths. The node can make its own decision to be an internal or a boundary node.

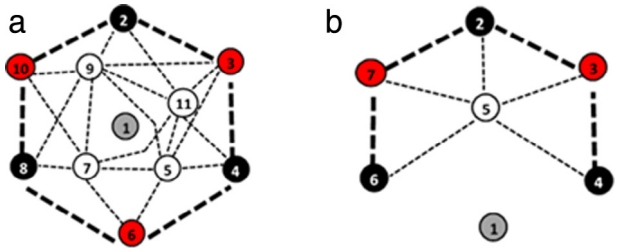


Fig. 2 – Connected independent set of cardinality $\alpha = 3$: (a) Closed path. (b) Broken path.

(1) *Gathering connectivity information*

Each node sends a "*Hello*" message. Thus, each node n_i receives a message from each of its neighbors and can create a list of its one-hop neighbors. This list is transmitted to its direct neighbors.

Upon receiving this list, each node can construct its one-hop neighbors' graph. The 2-hop neighbors are eliminated from the graph and the node keeps only its 1-hop neighbors. We note that the node n_i must also be removed from the graph.

(2) *Creating lists of independent sets*

Starting with the 1-hop neighbors' graph of a node n_i, the goal is to search for all independent sets having cardinality α. The simulation results showed that the optimal value is $\boldsymbol{\alpha = 3}$.

There exist many algorithms in literature for looking for the independent sets in a graph. The easiest way to find this set is the heuristic proposed in [14]. As mentioned before, a graph has more than a maximal independent set. Therefore, in this step, we search for all independent sets (ISs). The first is determined as follows: We start with the node having the minimum or maximum *id* in the graph G_i, which will be the first element of the set IS_1. Then we remove all the neighbors of this node from G_i. We choose another node having the minimum *id* among the remaining nodes of G_i, and this process is repeated until is remaining in G_i. This procedure is repeated to build all other possible ISs.

Fig. 2(a) depicts an example of the 1-hop neighbor's graph. The nodes 2, 3, 4, 5, 6, 7, 8, 9, 10 and 11 are the 1-hop neighbors of the node 1. The nodes 2, 4 and 8 form an independent set of cardinality 3.

(3) *Connection of independent sets*

To select the boundary nodes, the following rules should be used for each node:

R1: A node is internal if its 1-hop neighbors' graph has at least an independent set of cardinality α or more.

R2: To connect each two nodes of the same independent set, there should exist a set of nodes that are not neighbors of another node in the set.

Any node that does not satisfy both the rules is a boundary node. Otherwise, the node is selected to be an internal one. The BDCIS Algorithm's pseudo code is given below:

```
For each node n_i
α=3; /* the optimal value of α is 3*/
n_i.boundary = true; /* initially, all nodes are boundary nodes */
Begin
Send (N_1(n_i ))      /* send neighbors list to its neighbors */
Receive (N_1(n_j))  /* receive neighbors lists from its neighbors */
 Construct N_1(ni) graph (G_i);
 Remove N_2(ni) from (G_i);
 Compute all the independent sets IS in (G_i)
 /* check the existence of connected independent set */
 For each x ∈ IS(k):
Begin
  /* test the cardinality and the connectedness of the independent set*/
  If |x| = α Λ Is-Connected (G_i, x) then
n_i.boundary = false;   /* n_i becomes an internal node */
      Break;
end;
end.
```

```
Function Is-Connected (G_i, x)
For each pair of x:
Begin
  Pick two items (a, b) from the set x;
  /*remove the remaining items with their neighbors from the graph G*/;
  For all c ∈ x Λ c ∉ {a, b}: G = G-(c + N_1(c));
  If there exists a path from "a" to "b" in the graph G
  then the pair (a, b) is connected.
  end;
  If all pairs are connected then return True;
End.
```

For the connection of the independent sets, we choose two nodes of each set and then we remove the remaining node of the set with its neighbors. From the set of the remaining nodes, we select the minimum subset that connects the two chosen nodes. This procedure is repeated for each pair in the set.

Fig. 3 depicts a zoomed portion of the network captured from our simulator (presented in Section 4). The black and green circles represent the 1-hop neighbors of the node "1". In addition, the red circles represent the boundary nodes and the remaining black circles are internal nodes. The dashed lines are communication links between adjacent nodes.

In Fig. 3(a), the node "1" is an internal node. The nodes {7, 12, and 15} form an independent set of cardinality 3. We can see clearly that these nodes are well distributed in different sides of the node "1". To connect each two pairs of the set respecting the rules R1 and R2, we choose the pair (7, 12) and we exclude the node 15 with its neighbors {2, 3, 4, 5, 13, 14}. From the remaining nodes, the set {8, 9, and 10} is used to connect this pair. This procedure is repeated for the other pairs of the set (7, 15) and (12, 15). The connection process is represented by black lines which form a cycle around the node "1".

In Fig. 3(b), the node "1" belongs to the boundary of an obstacle (hole). After the execution of the first step of the BDCIS algorithm, the node "1" build its neighbor's graph as it is depicted in Fig. 3(b).

This graph contains the set of its 1-hop neighbors {2, 3, 4, 5, 6, 7, 8, 9, 10, 11} and the set of its 2-hop neighbors {12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33} as well as the connectivity between each item of the first set and its neighbors of the first and the second sets. We note that the nodes of the second set (2-hop neighbors) are used only in the connection procedure of each independent set.

After gathering connectivity information, the algorithm searches from all independent sets (ISs) of cardinality α ($\alpha = 3$). The possible ISs are {(2, 5, 9), (2, 5, 8),(2, 5, 10), (2, 5, 11), (3, 7, 9), (4, 7, 9)}.

We choose the IS (2, 5, 9) as depicted in Fig. 3(b). To connect each two pairs of this set according to the rule-2 defined before, we choose the pair (2, 5) and we exclude the node (9) with its neighbors. The remaining set is {3, 4, 7} from *set*-1. We note that we can use some elements of the *set*-2 (e.g. nodes 24, 27, 29, 30, 31) because there is connectivity between these nodes and the remaining nodes of the *set*-1. In our example the set {3, 4} is used to connect the pair (2, 5). We note that we can use other sets such as {29, 4} or {3, 27} to connect this set.

This procedure is repeated for the other pairs of the set (5, 9) and (2, 9). The connection links are represented by black lines. The pair (5, 9) is connected by the set {6, 8}, but we cannot find remaining nodes to connect the pair (2, 9). In this case, the path is broken and cannot form a closed cycle. Therefore, the node "1" sets itself as a boundary node.

4. Simulations and performances evaluation

To analyze the performances of our algorithm, we developed a simple simulator using Matlab tool R2013b. This Matlab version is dedicated to the simulation of computer networks protocols. In addition, it allows us to display the deduced boundary nodes (Figs. 3, 9, 11, and 13) in a simulated network. This is the reason why we did not use simulators such as NSx or OMNET++. In these experiments, 3500 nodes are randomly deployed in a rectangular region of 500 × 500 [m], and two holes of 50 m^2 are created by removing a portion of nodes in the center of the network [2]. Our algorithm is compared to three other algorithms: THD [10], SDBR [11] and BCP [12].

The remaining simulation parameters are described in details in Table 2.

4.1. *Performance evaluation*

At first, we examine the performance of the proposed algorithm BDCIS by varying the communication range (CR) between 14 (m) and 20 (m), we deduce then different average node degrees (Fig. 4).

While increasing CR, new nodes enter the communication area of each node and become its neighbors; therefore, the average node degree in the whole network will increase.

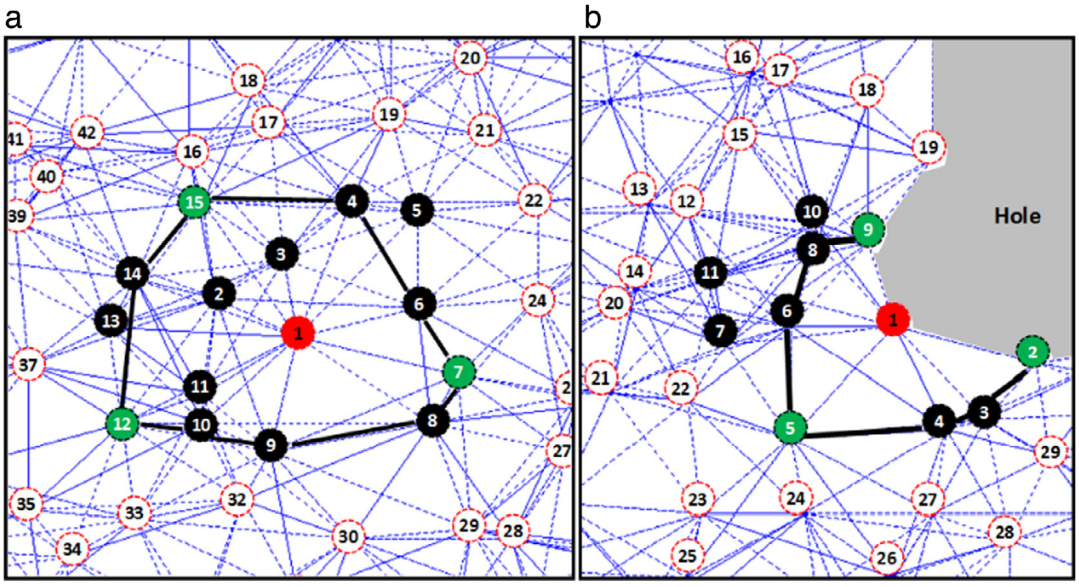


Fig. 3 – Connected independent sets of cardinality $\alpha = 3$. (a) Closed path. (b) Broken path.

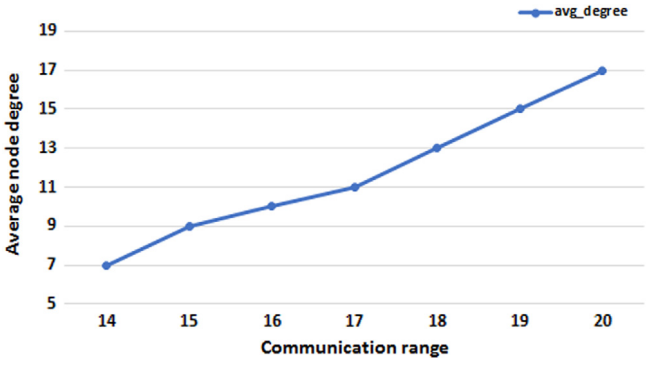


Fig. 4 – Average node degree according to the communication range.

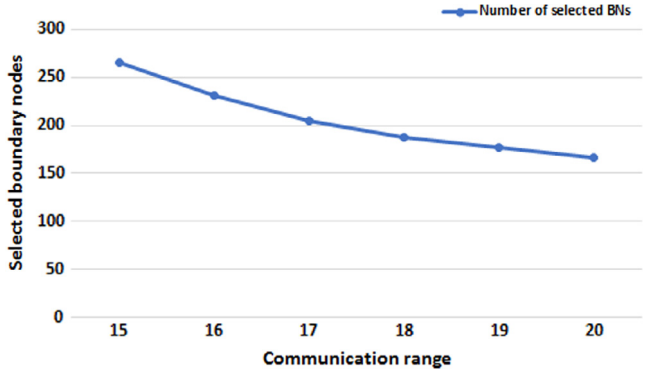


Fig. 5 – Average number of selected boundary nodes by BDCIS according to the communication range.

Fig. 5 shows the number of selected boundary nodes (BNs) according to the average node degree.

We can notice that the number of BNs decreases when increasing the average degree. When increasing CR, new nodes enter in the communication range of each node, therefore, the boundary will change (some boundary nodes become redundant BNs and are selected as internal nodes).

According to Fig. 6(a); the nodes A, B and C are boundary nodes. While increasing the communication range, the average node degree of the nodes A, B and C will increase

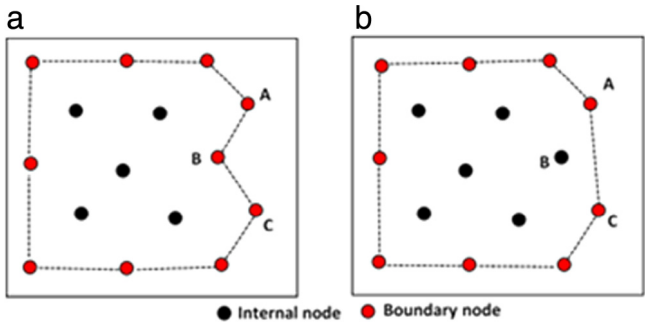


Fig. 6 – Example of the influence of the communication range on the selection of the boundary nodes.

Table 2 – Simulation parameters.

Zone (m)	500 × 500
Number of nodes	3500
Average degrees	07, 09, 11, 13, 15
Zone form	Rectangle
C_R (m)	13..19
Obstacles	2
Obstacle form	Rectangle
Hole size (m)	50
Energy consumed in sent message	1 Unit
Energy consumed in received message	0.20 Unit

(Fig. 6(b)). In this case the nodes "A" and "C" become neighbors and by consequence the node "B" becomes an internal node (it will be covered by the nodes A and C), thus it will not be selected as a "BN".

Fig. 7 will illustrate the number of exchanged messages.

We notice from Fig. 7 that the number of broadcast messages per node is constant regardless the average node degree in the network. However, the number of received messages increases proportionally to the average degree of nodes. Indeed, in the first step of the BDCS algorithm, each node needs to send two messages: one message for the neighbors' discovery and the other to transmit the list of neighbors. When a node broadcasts these two messages, all its neighbors receive them. Thus, the number of received messages will increase while increasing the average node degree.

In the following we will look for the accuracy ratio of our approach. Accuracy ratio is the value of the total number of boundary or internal nodes selected by our algorithm, divided by the value of total number of real boundary or internal nodes respectively [2]. To select the real boundary nodes, we implemented a simplified version of the CBNS algorithm presented in [5], where the nodes' positions are used to discover the real boundary.

Fig. 8 illustrates the accuracy ratio of the boundary nodes detection while varying the average node degree in the network. Our algorithm can detect most of boundary nodes (>95%) for different average node degrees. We can see that the detection rate of the boundary nodes decreases slightly while increasing the average node degree. This is due to the selection procedure of CBNS which selects always the first node encountered in its right side, where BDCIS search always for the closed paths.

In Fig. 9, the node "1" lies near the boundary, but it is *covered* by its neighbors 2, 3, and 4. So it is considered as an internal node by BDCIS, while CBNS recognizes this node as a boundary node. This problem of the boundary uniqueness is more detailed in [15]. If we reduce the communication range, the nodes "3" and "4" will not be neighbors, consequently the node "1" will not be covered and it will be easily recognized as a boundary node.

In Fig. 10 we will illustrate the false positive detection. The false positive detections are the internal nodes that are misclassified as boundary nodes.

Fig. 10 depicts the ratio of internal nodes misclassified as boundary nodes. In sparse networks with low node degrees, the network will not be completely covered, thus, many micro holes will appear. The nodes that surround these holes will be detected by our protocol as boundary nodes. In this case, the ratio of false detections is about 50%.

In Fig. 10, we zoomed a portion of the network after running the BDCIS algorithm. We can see clearly that the boundary nodes of the micro holes are marked by our algorithm.

When the average node degree increases, the network becomes well covered, and consequently the number of micro holes will decrease, which will reduce the number of false detections by our algorithm. In the dense networks with high node degrees, there will be no false detections.

We will now compute the energy consumed by our approach. Exchanged messages are used as an indication of energy consumption of our algorithm [16].

Fig. 12 shows the energy consumed in units by the BDCIS algorithm. The energy consumption increases proportionally to the average node degree. Indeed, when augmenting the number of neighbors of a node, the ratio of received messages will increase, this causes more energy consumption.

4.2. *Performance comparison*

At first, we give visual comparison of the four algorithms. By adjusting the communication range between 13 and 19 m, we can obtain different average node degrees varying between 7 and 15. The simulation results of the different algorithms are shown in Fig. 13(a), (d). The red stars represent the selected boundary nodes (BNs) and the blue dots represent internal nodes. (We note that for more clarity, nodes are distributed in a random grid).

When the average node degree is low (Fig. 13(a)), many micro holes appear in the network due to the insufficient

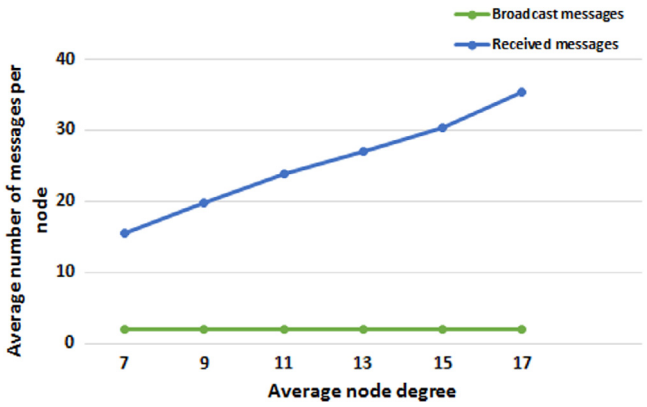


Fig. 7 – Average Number of exchanged messages per node.

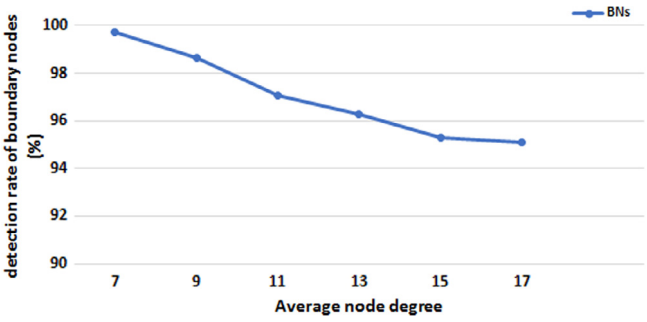


Fig. 8 – Detection ratio of boundary nodes according to the average node degree.

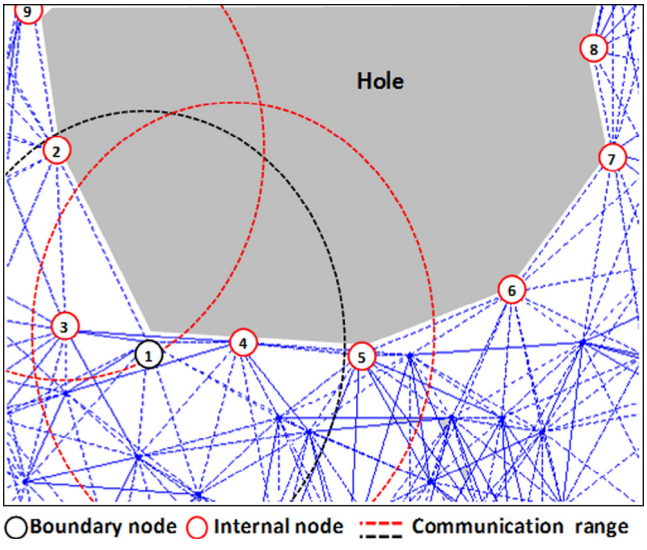


Fig. 9 – Boundary nodes missed by BDCIS.

coverage. The apparent noise is caused by these small holes, which are surrounded by marked nodes. All the algorithms find difficulties to distinguish between internal and boundary nodes.

When we increase the average node degree (Fig. 13(b)–(d)), the network becomes well covered and the number of micro holes decreases. Consequently, our algorithm will be able to distinguish clearly between boundary and internal nodes. Like BCP, our algorithm can always select a fine boundary, which is not the case with THD and SDBR that select a coarse boundary. We can see also that SDBR cannot distinguish between adjacent holes.

In the following, a quantitative evaluation of our approach will be described. It is made after several simulation runs of

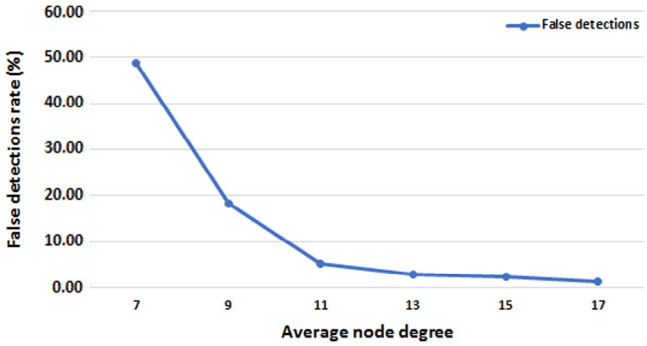


Fig. 10 – False detections ratio according to the average node degree.

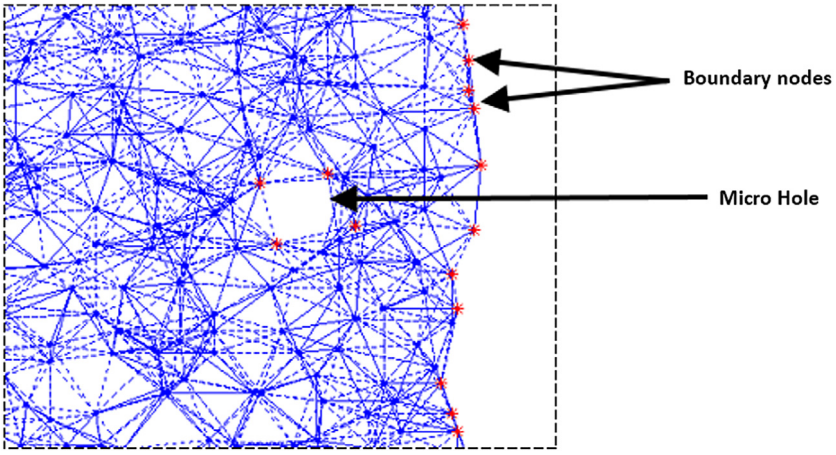


Fig. 11 – Zoomed portion of the network with average node degree (11).

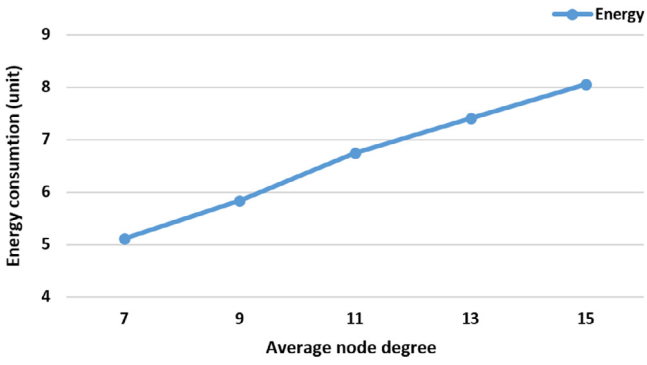


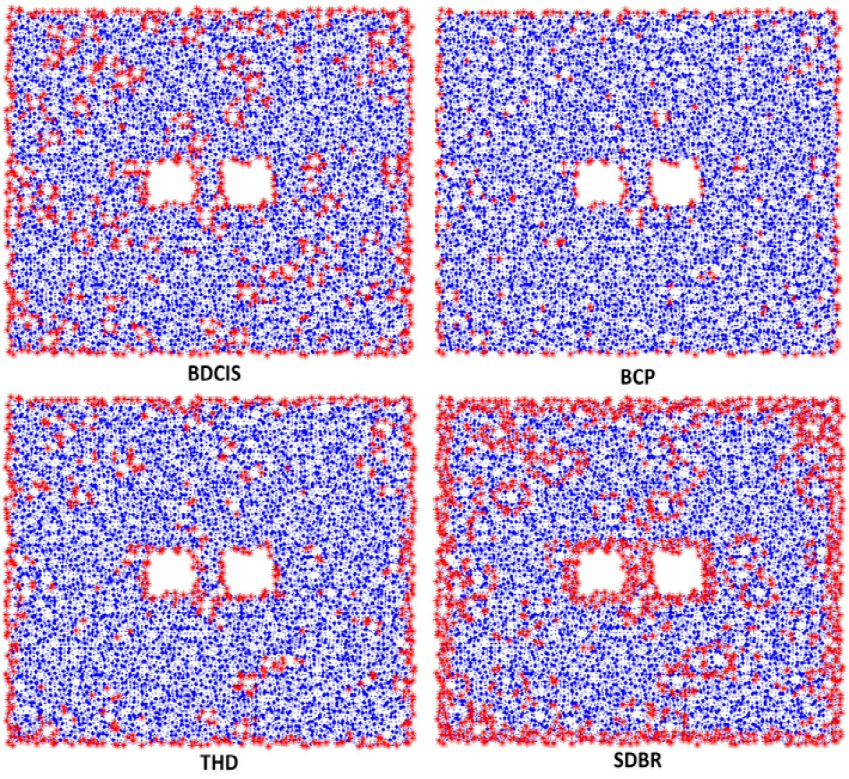
Fig. 12 – Energy consumed by BDCIS for boundary detection according to the average node degree.

the four algorithms (BDCIS, BCP, THD, and SBEDR) and only the average values are maintained.

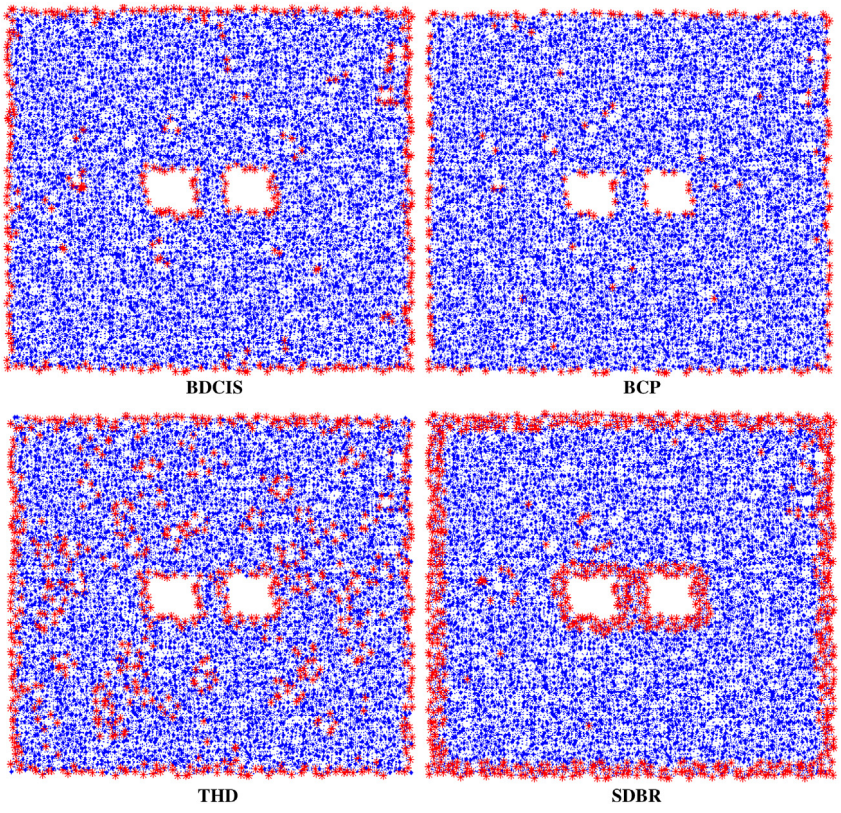
We evaluate here the detection rate of boundary nodes for each algorithm.

Fig. 14 illustrates the accuracy rate of the boundary nodes detection while varying the average node degree in the network. The SDBR algorithm produces a result close to our algorithm, while BCP and THD fail to detect a large number of boundary nodes. The SDBR algorithm selects all boundary nodes, because we did not take into account "micro holes". The hole is defined in [4] as a face which has four edges or more and cannot be divided in small faces.

In Fig. 15(a) "ABCD" is a micro-hole with four edges. So the nodes A, B, C and D must be selected as boundary nodes. The node "B" can find a closed path composed of its 2-hop neighbors (the green dashed line), so the SDBR algorithm selects the node "B" as internal node, which is considered as false detection of the boundary node. So, if the micro-holes are considered in the evaluation, the accuracy rate of SDBR will decrease when increasing the number of small holes.

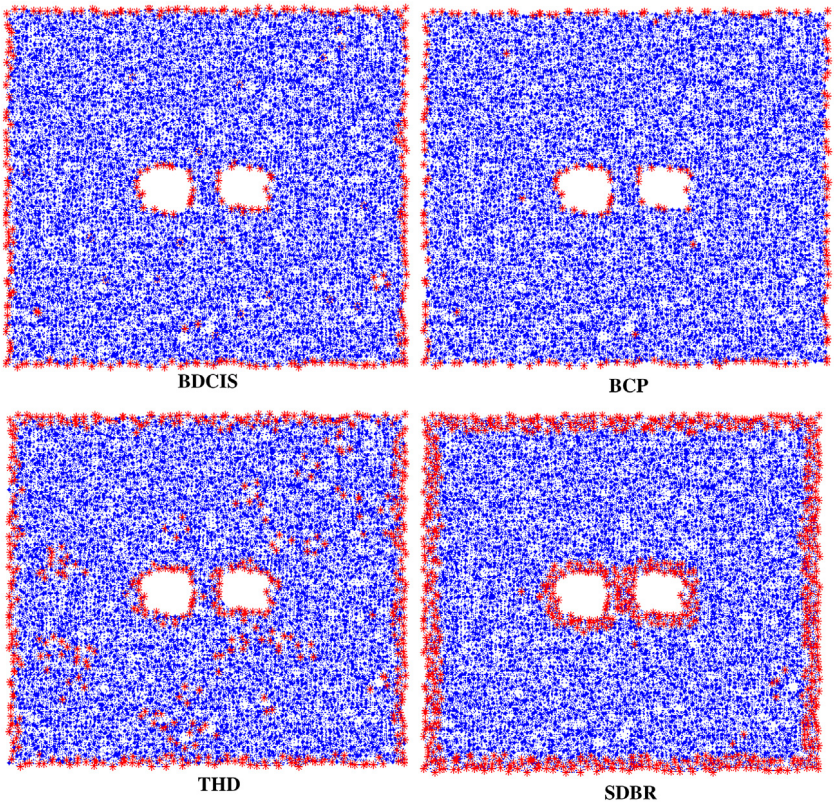


(a) Comparison of four algorithms with average node degree (9).

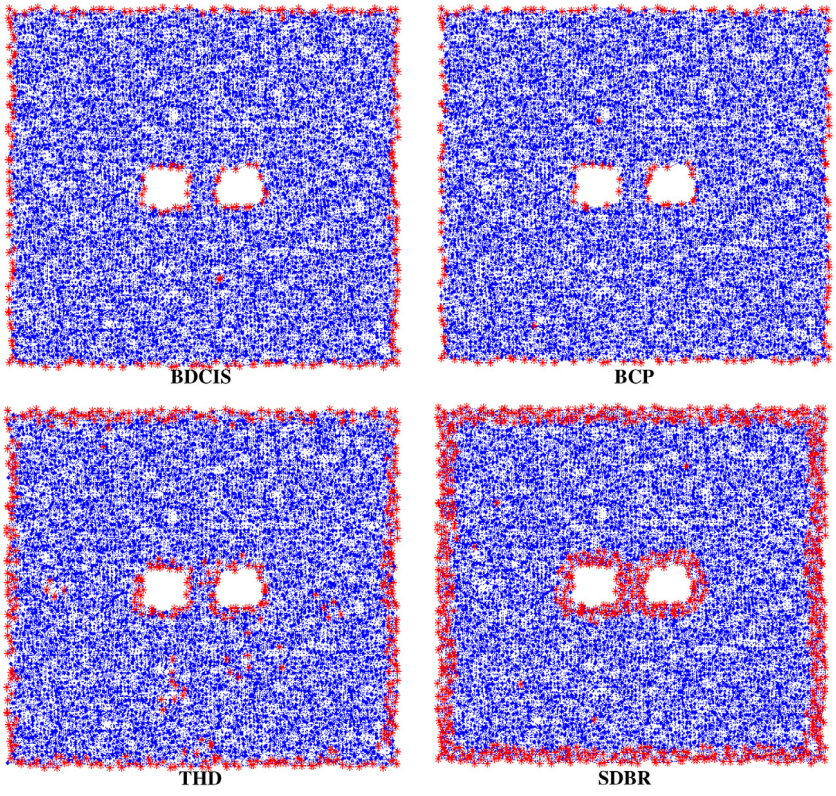


(b) Comparison of four algorithms with average node degree (11).

Fig. 13 – Comparing the four algorithms while varying node degree.



(c) Comparison of four algorithms with average node degree (13).



(d) Comparison of four algorithms with average node degree (15).

Fig. 13 – (*continued*)

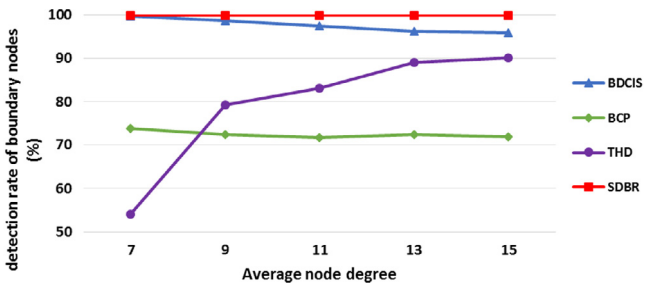


Fig. 14 – Detection ratio of boundary nodes while varying the average node degrees.

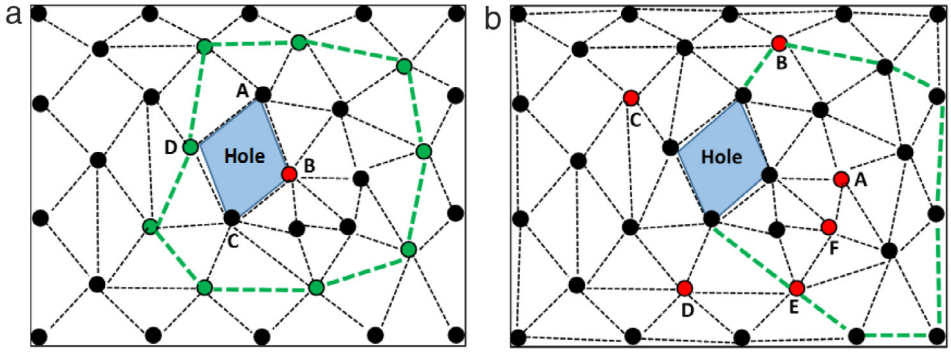


Fig. 15 – Misclassification of boundary nodes near the micro holes by SDBR (a) boundary nodes, (b) internal nodes.

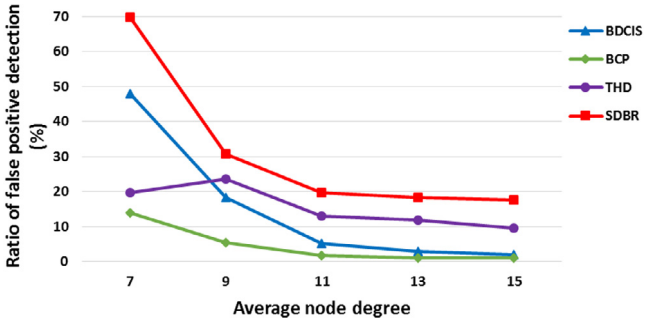


Fig. 16 – Ratio of false detections for different average node degrees.

We can see in Fig. 16 that the SDBR algorithm has the highest ratio of false detections because it selects the boundaries coarsely. Indeed, all nodes, which are 1-hop from the boundary, are selected as boundary nodes. The Fig. 15(b) depicts an example of this false detection. The node "A" is an internal node but its path of 2-hop neighbors (green line) is broken when it meets a small hole, so it is selected as a boundary node. The same conclusion while considering nodes {B, C, D, E, F}.

The BCP has the least ratio of false detections because it does not detect most of boundary nodes of the micro-holes. Even with high average node degrees, the THD algorithm has a relatively high false detections ratio because all nodes which have not "higher" 2-hop neighbors in their iso-lines are selected as boundary nodes.

We will now illustrate the overhead of each considered algorithm (see Fig. 17).

In BDCIS algorithm, each sensor node needs to send two messages: one message for the neighbors' discovery and another message containing the neighbors' list of each node. Therefore, the cost of this phase is $2N$ messages, where N is the number of sensors in the network.

In the SDBR algorithm, each node needs to send three messages to gather the connectivity information of its 2-hop neighbors. The THD algorithm generates the largest message overhead because it floods the whole network to select the seeds' set and to build iso-contours around each seed node. However, in BCP each node has to send one message to create its LVP. BCP has the less number of messages overhead because it uses the signal's angle of arrival message to extract topological information.

In this last experiment, we will evaluate the energy consumption of each algorithm (see Fig. 18).

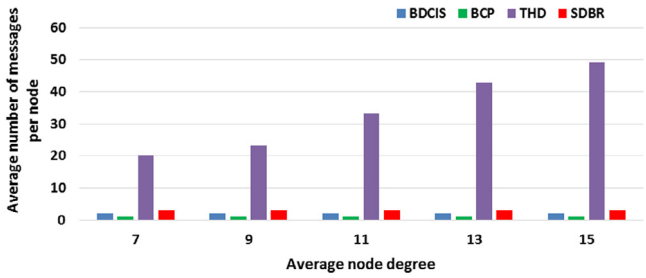


Fig. 17 – Control overhead for different average node degrees.

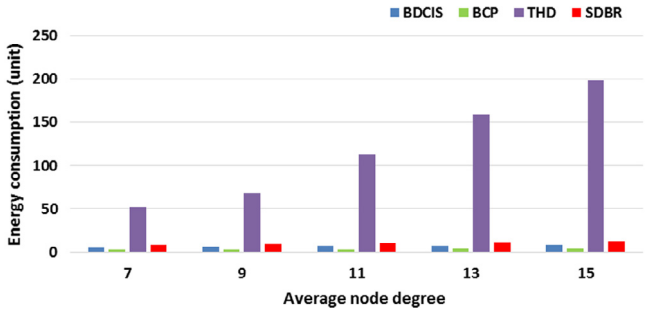


Fig. 18 – Energy consumption for different average node degrees.

The energy consumption is proportional to the average node degree. Indeed, each node broadcasts a message; this message will be received by its neighbors. Our algorithm consumes lower energy compared to THD and SDBR since it involves less messages exchange. BCP has the lowest energy consumption because each node needs only to receive one message from its direct neighbors to construct its LVP. Even BCP has the least energy consuming; it detects less boundary nodes than our algorithm.

5. Conclusion

We presented a distributed algorithm BDCIS for the boundary detection in wireless sensor networks where each node can take its own decision to be an internal or a boundary node relying solely on local connectivity information. First of all, we presented some algorithms proposed in the literature to overcome the problem of boundary detection in WSNs and we mentioned their classification into geometrical approaches, statistical approaches and topological approaches. Second, and as an alternative, we proposed a new protocol for the boundary detection in WSNs.

The main idea of BDCIS is based on the independent set's concept from the graph theory to select nodes on both boundaries of the holes and of the network. It is divided into three main steps. In the first step, each node collects connectivity information of its one-hop neighbors and constructs its one-hop neighbors' graph. In the second step, independent sets of cardinality α are established. In the last step, connecting sets and searching for closed paths. The node can make its own decision to be an internal or a boundary node.

To analyze and evaluate the performance of the proposed protocol, we developed a graphical simulator written in Matlab. The simulation results and the comparison to other works have proved that our proposed solution can detect network boundaries with high accuracy, low energy consumption and less communication overhead.

In our future works, our purpose will be the extension of the algorithm to networks having low density and also to heterogeneous networks. We aim also to connect these boundary nodes to obtain meaningful cycles, which will be exploited in many wireless sensor networks applications such as routing, positioning and target tracking.

REFERENCES

[1] M. Aissani, Routage temps-réel, tolérant aux vides et optimisant l'énergie dans les réseaux de capteurs sans fil, RCSFs, in: The Proceedings of the International Summer School, Bejaia, 2013.

[2] K.Y. Hsieh, J.P. Sheu, Hole detection and boundary recognition in wireless sensor networks, in: Personal, Indoor and Mobile Radio Communications, IEEE, 2009, pp. 27–76.

[3] Y. Wang, J. Gao, J.S.B. Mitchell, Boundary recognition in sensor networks by topological methods, in: Proc of the 12th Annual International Conference on Mobile Computing and Networking, Los angeles, California, 2006, pp. 122–133.

[4] Q. Fang, J. Gao, L. Guibas, Locating and bypassing routing holes in sensor networks, in: Proc. of Mobile Networks and Applications, vol. 11, 2006, pp. 187–200.

[5] P.K. Sahoo, J.P. Sheu, K.Y. Hsieh, Target tracking and boundary node selection algorithms of wireless sensor networks for internet services, Inform. Sci. 230 (1) (2013) 21–38.

[6] S.P. Fekete, M. Kaufmann, A. Kroller, K. Lehmann, A new approach for boundary recognition in geometric sensor networks, in: Proceedings of the 17th Canadian Conference on Computational Geometry, CCCG'05, Windsor, 2005, pp. 82–85.
[7] S.P. Fekete, A. Kroller, D. Pfisterer, S. Fischer, C. Buschmann, Neighborhood-based topology recognition in sensor networks, in: Proceedings of the 1st International Workshop on Algorithmic Aspects of Wireless Sensor Networks, Turku Finland, July 2004, vol. 3112, pp. 123–136.
[8] G. Destino, G.T. Freitas de Abreu, Network boundary recognition via graph-theory, in: Proceedings of the 5th Workshop on Positioning, Navigation and Communication, 2008, pp. 271–275.
[9] S. Funke, Topological hole detection in wireless sensor networks and its applications, in: Proc of the Joint Workshop on Foundations of Mobile Computing, 2005, pp. 44–53.
[10] S. Funke, C. Klein, Hole detection or: How much geometry hides in connectivity? in: The Proceedings of the 22nd ACM Symposium Computational Geometry, SCG'06, 2006, pp. 377–385.
[11] I.M. Khan, M.Z. Khan, H. Mokhtar, M. Merabti, Enhancements of the self-detection scheme for boundary recognition in wireless sensor networks, in: Developments in E-systems Engineering (DeSE), IEEE, 2011, pp. 448–453.
[12] A. Dabba, R. Beghdad, BCP: a border coverage protocol for wireless sensor networks, in: The Proceedings of IEEE Science and Information Conference SAI'2014, London, UK, pp. 632–640.
[13] F. Yan, P. Martins, L. Decreusefond, Connectivity-Based Distributed Coverage Hole Detection in Wireless Sensor Networks, IEEE, 2011, pp. 1–6.
[14] A. Khalil, R. Beghdad, Coverage and connectivity protocol for wireless sensor networks, in: The Proceedings of IEEE 24th International Conference on Microelectronics, ICM, Algiers, Algeria, 2012, pp. 1–4.
[15] O. Saukh, R. Sauter, M. Gauger, P.J. Marrón, On boundary recognition without location information in wireless sensor networks, ACM Trans. Sensor Netw. (TOSN) 6 (3) (2010) 20.
[16] W.C. Chu, K.F. Ssu, Location-free boundary detection in mobile wireless sensor networks with a distributed approach, Elsevier Comput. Netw. (70) (2014) 96–112.